

# Split-Brain Autoencoders: Unsupervised Learning by Cross-Channel Prediction

Richard Zhang

Phillip Isola

Alexei A. Efros

Berkeley AI Research (BAIR) Laboratory  
University of California, Berkeley

{rich.zhang, isola, efros}@eecs.berkeley.edu

## Abstract

We propose *split-brain autoencoders*, a straightforward modification of the traditional autoencoder architecture, for unsupervised representation learning. The method adds a split to the network, resulting in two disjoint sub-networks. Each sub-network is trained to perform a difficult task – predicting one subset of the data channels from another. Together, the sub-networks extract features from the entire input signal. By forcing the network to solve cross-channel prediction tasks, we induce a representation within the network which transfers well to other, unseen tasks. This method achieves state-of-the-art performance on several large-scale transfer learning benchmarks.

## 1. Introduction

A goal of unsupervised learning is to model raw data, without the use of labels, in a manner which produces a “useful” representation. Such a representation should be easily adaptable for other tasks, which are unknown during training time. Figure 1 (top) shows the autoencoder architecture, a commonly used method for unsupervised learning. We propose a straightforward modification, shown in Figure 1 (bottom), adding a split in the network architecture. The split forces the network to solve complementary prediction problems, and dramatically improves transfer performance.

Autoencoders [15] produce feature representations by reconstructing an input signal through a convolutional neural network (CNN). To prevent a trivial identity mapping from being learned, a bottleneck is typically built into the representation to force abstraction. However, an inherent tension is at play: the smaller the bottleneck, the greater the forced abstraction, but the smaller the information content that can be expressed. Autoencoders have been shown to produce relatively weak representations for transfer tasks

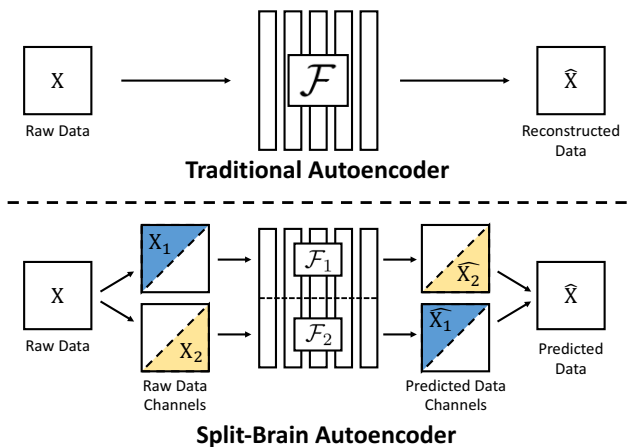


Figure 1: **Traditional vs Split-Brain Autoencoder architectures.** (top) Autoencoders learn feature representation  $\mathcal{F}$  by learning to reconstruct input data  $X$ . (bottom) The proposed split-brain autoencoder is composed of two disjoint sub-networks  $\mathcal{F}_1, \mathcal{F}_2$ , each trained to predict one data subset from another, changing the problem from reconstruction to prediction. The split-brain representation  $\mathcal{F}$  is formed by concatenating the two sub-networks, and achieves strong transfer learning performance. The model will be publicly released on <https://richzhang.github.io/splitbrainauto>.

on large and small-scale datasets alike [36, 28].

Instead of forcing abstraction through the network architecture, recent work has explored *withholding parts of the input*, or *input dropout* during training [36, 28, 42]. For example, Vincent et al. [36] propose denoising autoencoders, trained to remove *iid* noise added to the input. Pathak et al. [28] propose *context encoders*, trained to inpaint large, randomly dropped, contiguous blocks of pixels, and produce impressive graphics results. Several works [42, 23, 16] have trained networks to predict color data from grayscale, also known as the graphics task of colorization.

Method	No bottleneck required	Uses input dropout	No domain gap	No input handicap
Autoencoder [15]	×	×	✓	✓
Denoising autoencoder [36]	✓	✓	×	✓
Context Encoder [28]	✓	✓	×	✓
Cross-Channel Encoder [42]	✓	✓	✓	×
Split-Brain Autoencoder	✓	✓	✓	✓

Table 1: **Qualitative Comparison of Various Representation Learning Techniques** Characters ✓ and × indicate a desired and undesired characteristic, respectively, inherent in the training methodology. **No bottleneck required:** no architectural bottleneck is required to force abstraction; in other words, an identity mapping cannot solve the problem. **Uses input dropout:** method uses some form of input dropout for training. **No domain gap:** no gap between the input data during pre-training and testing time. **No input handicap:** the method is not handicapped at test time; no input data is systematically dropped out during test time

A qualitative comparison of the different methods, along with their inherent strengths and weaknesses, is summarized in Table 1. Context encoders, though stronger than traditional autoencoders, demonstrate lower performance than competitors on large-scale representation learning benchmarks [42]. This may be due to two reasons. Firstly, image synthesis tasks are known to be notoriously difficult to evaluate [29] and the loss function used in [28] may not properly capture inpainting quality. Secondly, the model is trained on images with missing chunks, but applied, at test time, to full images. This causes a “domain gap” between training and deployment.

Interestingly, colorization serves as an effective pretext task for inducing strong feature representations [42, 23]. This may be possible because (i) though colorization, like inpainting, is a synthesis task, the spatial correspondence between the input and output pairs enables basic off-the-shelf loss functions to be effective and (ii) the *systematic*, rather than *stochastic* nature of the input corruption removes the pre-training and testing domain gap. Colorization is an example of what we refer to as a *cross-channel encoding* objective, a task which directly predicts one subset of data channels from another. In this work, we further explore the space of cross-channel encoders by systematically evaluating various channel translation problems and training objectives. Cross-channel encoders, however, face an inherent handicap: different channels of the input data are not treated equally, as part of the data is used for feature extraction and another is used as a prediction target. In the case of colorization, the network can only extract features from the grayscale image, and is blind to the color channels, leaving the color information unused.

Might there be a way to take advantage of the underlying principle of cross-channel encoders, while being able to

extract features from the entire input signal? We propose an architectural modification to the autoencoder paradigm: adding a single split in the network, resulting in two disjoint, concatenated, sub-networks. Each sub-network is trained as a cross-channel encoder, predicting one subset of channels of the input from the other. For example, on RGB images, one sub-network solves the problem of colorization (predicting  $a$  and  $b$  channels from the  $L$  channel in  $Lab$  colorspace), and the other performs the opposite (synthesizing  $L$  from  $a, b$  channels). In the RGB-D domain, one sub-network predicts depth from images, and the other predicts images from depth. The full network is then able to extract features from the full input tensor, leaving nothing on the table. The architectural change induces the same forced abstraction as observed in cross-channel encoders.

Our contributions are as follows:

- We propose the split-brain autoencoder, which is composed of concatenated cross-channel encoders, trained using *raw data* as its own supervisory signal.
- We demonstrate state-of-the-art performance on several representation learning benchmarks in the RGB and RGB-D domains.
- To gain a better understanding, we perform extensive ablation studies by (i) investigating cross-channel prediction problems and loss functions and (ii) researching alternative aggregation methods for combining cross-channel encoders.

## 2. Related Work

Many unsupervised learning methods have focused on modeling raw data using a reconstruction objective. Autoencoders [15] train a network to reconstruct an input image, using a representation bottleneck to force abstraction. Denoising autoencoders [36] train a network to undo a random *iid* corruption. Techniques for modeling the probability distribution of images in deep frameworks have also been explored. For example, variational autoencoders (VAEs) [20] employ a variational Bayesian approach to modeling the data distribution. Other probabilistic models include restricted Boltzmann machines (RBMs) [34], deep Boltzmann machines (DBMs) [32], generative adversarial networks (GANs) [12], autoregressive models (Pixel-RNN [35] and Pixel-CNN [26]), bidirectional GANs (BiGANs) [5] and Adversarially Learned Inference (ALI) [6], and real NVP [3]. Many of these methods [15, 36, 5, 6, 32] have been evaluated for representation learning.

In our work, we show that several cross-channel prediction, or pixel-to-pixel translation problems, can be used for representation learning in the cross-channel encoding framework, even with basic off-the-shelf networks and objective functions. These tasks have been explored in great detail in previous works, with the emphasis of generating

strong results on the specific tasks themselves. For example, networks trained on RGB images have been shown to perform well in tasks such as image segmentation [40], semantic segmentation [24], image colorization [42, 23, 16], depth prediction [7], surface normal prediction [7, 37], and image stylization [19]. Previous work has also explored the task of generating raw images, conditioned on surface normals [39], text captions [30] and random latent noise vectors [12, 5].

Another form of unsupervised learning, sometimes referred to as “self-supervised” learning, has recently grown in popularity [1, 18, 4, 25, 38, 28, 17, 27]. Rather than predicting labels annotated by humans, these methods predict pseudo-labels, which are functions automatically computed from the raw data itself. For example, Doersch et al. [4] predict the relative position between two patches. Other methods generate pseudo-labels from egomotion [1, 18], video tracking [38], inpainting [28], co-occurrence [17], puzzle-solving [25], color [42, 23], and sound [27]. These methods generally focus on a single supervisory signal, and involve some engineering effort. In this work, we show that simply predicting raw data channels with standard loss functions is surprisingly effective, often outperforming these previously proposed self-supervised methods.

### 3. Methods

In Section 3.1, we define the paradigm of cross-channel encoding. In Section 3.2, we propose the split-brain autoencoder and explore alternative methods for aggregating multiple cross-channel encoders into a single network.

#### 3.1. Cross-Channel Encoders

We would like to learn a deep representation on input data tensor  $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$ , with  $C$  channels. We split the data into  $\mathbf{X}_1 \in \mathbb{R}^{H \times W \times C_1}$  and  $\mathbf{X}_2 \in \mathbb{R}^{H \times W \times C_2}$ , where  $C_1, C_2 \subseteq C$ , and then train a deep representation to solve the prediction problem  $\widehat{\mathbf{X}}_2 = \mathcal{F}(\mathbf{X}_1)$ . Function  $\mathcal{F}$  is learned with a CNN, which produces a layered representation of input  $\mathbf{X}_1$ , and we refer to each layer  $l$  as  $\mathcal{F}^l$ . By performing this *pretext* task of predicting  $\mathbf{X}_2$  from  $\mathbf{X}_1$ , we hope to achieve a representation  $\mathcal{F}(\mathbf{X}_1)$  which contains higher-level abstractions or semantics.

This prediction task can be trained with various loss functions, and we study whether the loss function affects the quality of the learned representation. To begin, we explore the use of  $\ell_2$  regression, as shown in Equation 1.

$$\ell_2(\mathcal{F}(\mathbf{X}_1), \mathbf{X}_2) = \frac{1}{2} \sum_{h,w} \|\mathbf{X}_{2,h,w} - \mathcal{F}(\mathbf{X}_1)_{h,w}\|_2^2 \quad (1)$$

We also study the use of a classification loss. Here, the target output  $\mathbf{X}_2 \in \mathbb{R}^{H \times W \times C_2}$  is encoded with function  $\mathcal{H}$  into a *distribution*  $\mathbf{Y}_2 \in \Delta^{H \times W \times Q}$ , where  $Q$  is the

number of elements in the quantized output space. Network  $\mathcal{F}$  is then trained to predict a distribution,  $\widehat{\mathbf{Y}}_2 = \mathcal{F}(\mathbf{X}_1) \in \Delta^{H \times W \times Q}$ . A standard cross-entropy loss between the predicted and ground truth distributions is used, as shown Equation 2.

$$\ell_{cl}(\mathcal{F}(\mathbf{X}_1), \mathbf{X}_2) = - \sum_{h,w} \sum_q \mathcal{H}(\mathbf{X}_2)_{h,w,q} \log(\mathcal{F}(\mathbf{X}_1)_{h,w,q}) \quad (2)$$

In [42], the authors discover that classification loss is more effective for the graphics task of automatic colorization than regression. We hypothesize that for some tasks, especially those with inherent uncertainty in the prediction, the classification loss may lead to better representations as well, as the network will be incentivized to match the whole distribution, and not only predict the first moment.

Note that with input and output sets  $C_1, C_2 = C$ , and an  $\ell_2$  regression loss, the objective becomes identical to the autoencoder objective.

#### 3.2. Split-Brain Autoencoders as Aggregated Cross-Channel Encoders

We can train multiple cross-channel encoders,  $\mathcal{F}_1, \mathcal{F}_2$ , on opposite prediction problems, with loss functions  $L_1, L_2$ , respectively, described in Equation 3.

$$\begin{aligned} \mathcal{F}_1^* &= \arg \min_{\mathcal{F}_1} L_1(\mathcal{F}_1(\mathbf{X}_1), \mathbf{X}_2) \\ \mathcal{F}_2^* &= \arg \min_{\mathcal{F}_2} L_2(\mathcal{F}_2(\mathbf{X}_2), \mathbf{X}_1) \end{aligned} \quad (3)$$

By concatenating the representations layer-wise,  $\mathcal{F}^l = \{\mathcal{F}_1^l, \mathcal{F}_2^l\}$ , we achieve a representation  $\mathcal{F}$  which is pre-trained on full input tensor  $\mathbf{X}$ . Example split-brain autoencoders in the image and RGB-D domains are shown in Figures 2(a) and (b), respectively. If  $\mathcal{F}$  is a CNN of a desired fixed size, e.g., AlexNet [22], we can design the sub-networks  $\mathcal{F}_1, \mathcal{F}_2$  by splitting each layer of the network  $\mathcal{F}$  in half, along the channel dimension. Concatenated representation  $\mathcal{F}$  will then have the appropriate dimensionality, and can be simply implemented by setting the `group` parameter to 2 in most deep learning libraries.

As each channel in the representation is only connected to half of the channels in the preceding layer, the number of parameters in the network is actually halved, relative to a full network.

Note that the input and the output to the network  $\mathcal{F}$  is the full input  $\mathbf{X}$ , the same as an autoencoder. However, due to the split nature of the architecture, the network  $\mathcal{F}$  is trained to *predict*  $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2\}$ , rather than simply *reconstruct* it from the input. In essence, an architectural change in the autoencoder framework induces the same forced abstraction achieved by cross-channel encoding.

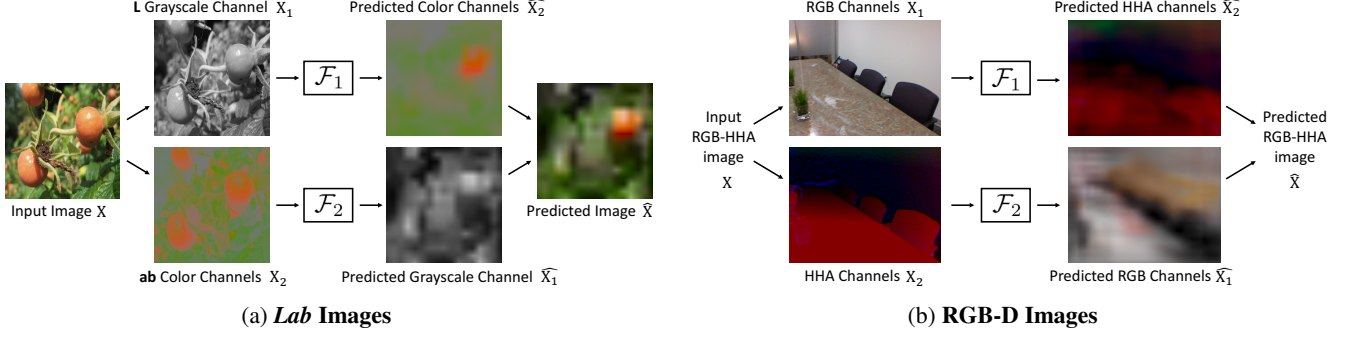


Figure 2: **Split-Brain Autoencoders applied to various domains** (a) **Lab images** Input images are divided into the  $L$  channel, which contains grayscale information, and the  $a$  and  $b$  channels, which contain color information. Network  $\mathcal{F}_1$  performs automatic colorization, whereas network  $\mathcal{F}_2$  performs grayscale prediction. (b) **RGB-D images** Input data  $\mathbf{X}$  contains registered RGB and depth images. Depth images are encoded using the HHA encoding [14]. Image representation  $\mathcal{F}_1$  is trained by predicting HHA channels. Representation  $\mathcal{F}_2$  on HHA images is learned by predicting images in  $Lab$  space. Note that the goal of performing these synthesis tasks is to induce representations  $\mathcal{F}_1, \mathcal{F}_2$  that transfer well to other tasks.

### 3.2.1 Alternative Aggregation Technique

We found the split-brain autoencoder, which aggregates cross-channel encoders through concatenation, to be more effective than several alternative strategies. As a baseline, we also explore an alternative aggregation strategy: the same representation  $\mathcal{F}$  can be trained to perform both mappings from  $\mathbf{X}_1$  to  $\mathbf{X}_2$ , and vice-versa. The loss function is described in Equation 4, with a slight abuse of notation: here, we redefine  $\mathbf{X}_1$  to be the same shape as original input  $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$ , with channels in set  $C \setminus C_1$  zeroed out. We make the analogous modification to  $\mathbf{X}_2$  as well.

$$\mathcal{F}^* = \arg \min_{\mathcal{F}} L_1(\mathcal{F}(\mathbf{X}_1), \mathbf{X}_2) + L_2(\mathbf{X}_1, \mathcal{F}(\mathbf{X}_2)) \quad (4)$$

The system only sees data subsets  $\mathbf{X}_1, \mathbf{X}_2$ , and never the full input  $\mathbf{X}$ . To alleviate this problem, we mix in the autoencoder objective as well, as shown in Equation 5, with  $\lambda \in [0, \frac{1}{2}]$ .

$$\begin{aligned} \mathcal{F}^* = \arg \min_{\mathcal{F}} & \lambda L_1(\mathcal{F}(\mathbf{X}_1), \mathbf{X}_2) + \lambda L_2(\mathcal{F}(\mathbf{X}_2), \mathbf{X}_1) \\ & + (1 - 2\lambda) L_3(\mathbf{X}, \mathcal{F}(\mathbf{X})) \end{aligned} \quad (5)$$

Note that unlike the split-brain architecture, in these objectives, there is a domain gap between the distribution of pre-training data and the full input tensor  $\mathbf{X}$ .

## 4. Experiments

In Section 4.1, we apply our proposed split-brain autoencoder architecture to learn unsupervised representations on large-scale image data from ImageNet [31]. We evaluate on established representation learning benchmarks and demonstrate state-of-the-art performance relative to previous unsupervised methods [21, 4, 38, 28, 27, 5]. In Section 4.2, we

apply the proposed method on the NYU-D dataset [33], and show performance above baseline methods.

### 4.1. Split-Brain Autoencoders on Images

We work with image data  $\mathbf{X}$  in the  $Lab$  color space, and learn cross-channel encoders with  $\mathbf{X}_1$  representing the  $L$ , or lightness channel, and  $\mathbf{X}_2$  containing the  $ab$  channels, or color information. This is a natural choice as (i) networks such as Alexnet, trained with grouping in their architecture, naturally separate into grayscale and color [22] even in a fully-supervised setting, and (ii) the individual cross-channel prediction problem of colorization,  $L$  to  $ab$ , has produced strong representations [42, 23]. In preliminary experiments, we have also explored different cross-channel prediction problems in other color spaces, such as  $RGB$  and  $YUV$ . We found the  $L$  and  $ab$  to be most effective split of the data.

To enable comparisons to previous unsupervised techniques, all of our trained networks use AlexNet architectures [22]. Because we are training for a pixel-prediction task, we run the network fully convolutionally [24]. The network is trained with random  $180 \times 180$  crops and predicts values at a heavily downsampled  $12 \times 12$  resolution. One can add upsampling-convolutional layers or use *atrous*[2]/*dilated*[41] convolutions to predict full resolution images at the expense of additional memory and run-time, but we found predicting at a lower resolution to be sufficient for representation learning. We pre-train on the 1.3M ImageNet dataset [31] (without the use of labels).

We train the following aggregated cross-channel encoders, including our primary method, the split-brain autoencoder.

- **Split-Brain Autoencoder (cl,cl) (Our full method):** A split-brain autoencoder, with one half performing colorization, and the other half performing grayscale

prediction. The top-level architecture is shown in Figure 2(a). Both sub-networks are trained for classification (cl), with a cross-entropy objective. In Figure 2(a), the point estimate made from the per-pixel predicted output distributions is annealed-mean from [42].

- **Split-Brain Autoencoder (reg,reg)**: Same as above, with both sub-networks trained with an  $\ell_2$  loss (reg).
- **Ensembled L→ab**: Two concatenated disjoint sub-networks, both performing colorization (predicting  $ab$  from  $L$ ). One subnetwork is trained with a classification objective, and the other with regression.
- **(L,ab)→(ab,L)**: A single network trained to perform both colorization and grayscale prediction, both with regression losses, as described in Equation 4. This baseline explores an alternative method for combining cross-channel encoders.
- **(L,ab,Lab)→(ab,L,Lab)**: Objective described in Equation 5. We set  $\lambda = \frac{1}{3}$ .

Single cross-channel encoders are ablations of our main method. We systematically study combinations of loss functions and cross-channel prediction problems.

- **L→ab(reg)**: Automatic colorization using an  $\ell_2$  loss.
- **L→ab(cl)**: Automatic colorization using a classification loss. We follow the quantization procedure proposed in [42]. The output  $ab$  space is binned into grid size  $10 \times 10$ , with a classification loss over the 313 bins that are within the  $ab$  gamut. Zhang et al. [42] use a class-rebalancing term, to over-sample rare colors in the training set, and a soft-encoding scheme for  $\mathcal{H}$ . Our objective is more straightforward, as we do not use class-rebalancing. In addition, we use a 1-hot encoding representation of classes rather than soft-encoding.
- **ab→L(reg)**: Grayscale prediction using an  $\ell_2$  loss.
- **ab→L(cl)**: Grayscale prediction using a classification loss. The  $L$  channel, which has values between 0 and 100, is quantized into 50 bins of size 2 and encoded.
- **Lab→Lab**: Autoencoder objective, reconstructing Lab from itself using an  $\ell_2$  regression loss, with the same architecture as the cross-channel encoders.
- **Lab(drop50)→Lab**: Same as above, with 50% of the input randomly dropped out during pre-training. This is similar to denoising autoencoders [36].

We compare to the following methods, which all use variants of Alexnet [22]. For additional details, refer to Table 3 in [42]. Note that one of these modifications resulted in a large deviation in feature map size<sup>1</sup>.

<sup>1</sup>The method from [25] uses stride 2 instead of 4 in the conv1 layer, resulting in  $4 \times$  denser feature maps throughout all convolutional layers. It is unclear how this change affects representational quality. The network uses the same number of parameters, but  $5.6 \times$  the memory and  $7.4 \times$  the run-time.

Task Generalization on ImageNet Classification [31]					
Method	conv1	conv2	conv3	conv4	conv5
ImageNet-labels [22]	19.3	36.3	44.2	48.3	50.5
Gaussian	11.6	17.1	16.9	16.3	14.1
Krähenbühl et al. [21]	17.5	23.0	24.5	23.2	20.6
<sup>1</sup> Noroozi & Favaro [25]	19.2	30.1	34.7	33.9	28.3
Doersch et al. [4]	16.2	23.3	30.2	31.7	29.6
Donahue et al. [5]	<b>17.7</b>	24.5	31.0	29.9	28.0
Pathak et al. [28]	14.1	20.7	21.0	19.8	15.5
Zhang et al. [42]	12.5	24.5	30.4	31.5	30.3
Lab→Lab	12.9	20.1	18.5	15.1	11.5
Lab(drop50)→Lab	12.1	20.4	19.7	16.1	12.3
L→ab(cl)	12.7	26.2	33.0	33.3	31.8
L→ab(reg)	12.3	23.5	29.6	31.1	30.1
ab→L(cl)	11.6	19.2	22.6	21.7	19.2
ab→L(reg)	11.5	19.4	23.5	23.9	21.7
(L,ab)→(ab,L)	15.1	22.6	24.4	23.2	21.1
(L,ab,Lab)→(ab,L,Lab)	15.4	22.9	24.0	22.0	18.9
Ensembled L→ab	11.7	23.7	30.9	32.2	31.3
Split-Brain Auto (reg,reg)	17.4	27.9	33.6	34.2	32.3
Split-Brain Auto (cl,cl)	<b>17.7</b>	<b>29.3</b>	<b>35.4</b>	<b>35.2</b>	<b>32.8</b>

Table 2: **Task Generalization on ImageNet Classification**

To test unsupervised feature representations, we train linear logistic regression classifiers on top of each layer to perform 1000-way ImageNet classification, as proposed in [42]. All weights are frozen and feature maps spatially resized to be  $\sim 9000$  dimensions. All methods use AlexNet variants [22], and were pre-trained on ImageNet without labels, except for **ImageNet-labels**. Note that the proposed split-brain autoencoder achieves the best performance on all layers across unsupervised methods.

Dataset & Task Generalization on Places Classification [43]					
Method	conv1	conv2	conv3	conv4	conv5
Places-labels [43]	22.1	35.1	40.2	43.3	44.6
ImageNet-labels [22]	22.7	34.8	38.4	39.4	38.7
Gaussian	15.7	20.3	19.8	19.1	17.5
Krähenbühl et al. [21]	21.4	26.2	27.1	26.1	24.0
<sup>1</sup> Noroozi & Favaro [25]	23.0	32.1	35.5	34.8	31.3
Doersch et al. [4]	19.7	26.7	31.9	32.7	30.9
Wang & Gupta [38]	20.1	28.5	29.9	29.7	27.9
Owens et al. [27]	19.9	29.3	32.1	28.8	29.8
Donahue et al. [5]	<b>22.0</b>	28.7	31.8	31.3	29.7
Pathak et al. [28]	18.2	23.2	23.4	21.9	18.4
Zhang et al. [42]	16.0	25.7	29.6	30.3	29.7
L→ab(cl)	16.4	27.5	31.4	32.1	30.2
L→ab(reg)	16.2	26.5	30.0	30.5	29.4
ab→L(cl)	15.6	22.5	24.8	25.1	23.0
ab→L(reg)	15.9	22.8	25.6	26.2	24.9
Split-Brain Auto (cl,cl)	21.3	<b>30.7</b>	<b>34.0</b>	<b>34.1</b>	<b>32.5</b>

Table 3: **Dataset & Task Generalization on Places Classification**

We train logistic regression classifiers on top of frozen pre-trained representations for 205-way Places classification. Note that our split-brain autoencoder achieves the best performance among unsupervised learning methods from conv2–5 layers.

- **ImageNet-labels** [22]: Trained on ImageNet labels for the classification task in a fully supervised fashion.
- **Gaussian**: Random Gaussian initialization of weights.
- **Krähenbühl et al.** [21]: A stacked k-means initialization method.
- **Doersch et al.** [4], **Noroozi & Favaro** [25], **Pathak et al.** [28], **Donahue et al.** [5], and **Zhang et al.** [42] all pre-train on the 1.3M ImageNet dataset [31].
- **Wang & Gupta** [38] and **Owens et al.** [27] pre-train on other large-scale data.

#### 4.1.1 Transfer Learning Tests

How well does the pre-text task of cross-channel prediction generalize to unseen tasks and data? We run various established large-scale representation learning benchmarks.

**ImageNet** [22] As proposed in [42], we test the *task* generalization of the representation by freezing the weights and training multinomial logistic regression classifiers on top of each layer to perform 1000-way ImageNet classification. Note that each classifier is a single learned linear layer, followed by a softmax. To reduce the effect of differences in feature map sizes, we spatially resize feature maps through bilinear interpolation, so that the flattened feature maps have approximately equal dimensionality (9600 for conv1, 3, 4 and 9216 for conv2, 5). The results are shown in Table 2 and Figures 3(a) and 4.

**Places** [43] In the previous test, we evaluated the representation on the same input training data, the ImageNet dataset, with a different task than the pretraining tasks. To see how well the network generalizes to new input *data* as well, we run the same linear classification task on the large-scale Places dataset [43]. The dataset contains 2.4M images for training and 20.5k for validation from 205 scene categories. The results are shown in Table 3 and Figure 3(b).

**PASCAL** [8] To further test generalization, we fine-tune the learned representation on standard representation learning benchmarks on the PASCAL dataset, as shown in Table 4, using established testing frameworks in classification [21], detection [11], and segmentation [24]. Classification involves 20 binary classification decisions, regarding the presence or absence of 20 object classes. Detection involves drawing an accurately localized bounding box around any objects in the image, and is performed using the Fast R-CNN [11] framework. Segmentation is pixel-wise labeling of the object class, either one of the 20 objects of interest or background. Here, the representation is *fine-tuned* through multiple layers of the network, rather than frozen. Prior to fine-tuning, we follow standard practice and use the rescaling method from [21], which rescales the weights so that the layers learn at the same “rate”, using the ratio of expected gradient magnitude over feature activation magnitude as a heuristic.

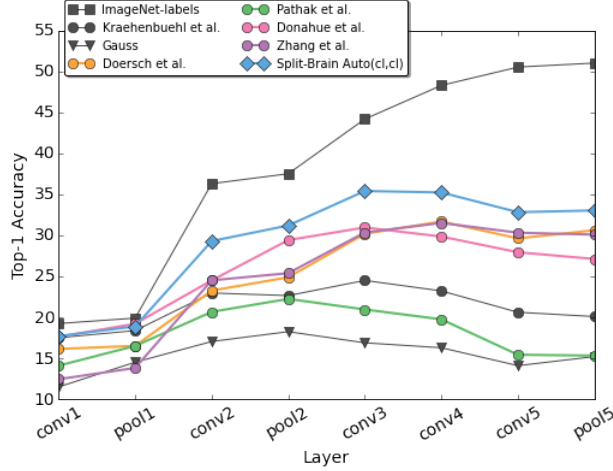
Task and Data Generalization on PASCAL VOC [8]						
	Classification [21] (%mAP)			Detection [11] (%mAP)		Seg. [24] (%mIU)
	Ref	conv5 fc6-8	none all	Ref	none all	Ref
<b>frozen layers</b>						
<b>fine-tuned layers</b>						
ImageNet labels [22]	[42]	78.9	79.9	[21]	56.8	[24] 48.0
Gaussian	[28]	–	53.3	[28]	43.4	[28] 19.8
Autoencoder	[5]	16.0	53.8	[28]	41.9	[28] 25.2
Krähenbühl et al. [21]	[5]	39.2	56.6	[21]	45.6	[5] 32.6
Jayaraman & Grauman [18]	–	–	–	[18]	41.7	–
Agrawal et al. [1]	[21]	–	52.9	[21]	41.8	–
Agrawal et al. [1] <sup>†</sup>	[5]	31.0	54.2	[21]	43.9	–
Wang & Gupta [38]	[21]	–	62.8	[21]	47.4	–
Wang & Gupta [38] <sup>†</sup>	[21]	–	63.1	[21]	47.2	–
Doersch et al. [4]	[21]	–	55.3	[21]	46.6	–
Doersch et al. [4] <sup>†</sup>	[5]	55.1	65.3	[21]	<b>51.1</b>	–
Pathak et al. [28]	[28]	–	56.5	[28]	44.5	[28] 29.7
Donahue et al. [5] <sup>†</sup>	[42]	50.2	58.6	[5]	46.2	[5] 34.9
Owens et al. [27]	▷	54.6	54.4	[27]	44.0	–
Owens et al. [27] <sup>†</sup>	▷	52.3	61.3	–	–	–
Zhang et al. [42] <sup>†</sup>	[42]	61.5	65.9	[42]	46.9	[42] 35.6
Split-Brain Auto (cl,cl) <sup>†</sup>	▷	<b>63.0</b>	<b>67.1</b>	▷	46.7	▷ <b>36.0</b>

Table 4: **Task and Dataset Generalization on PASCAL VOC** Classification and detection on PASCAL VOC 2007 [9] and segmentation on PASCAL VOC 2012 [10], using mean average precision (mAP) and mean intersection over union (mIU) metrics for each task, with publicly available testing frameworks from [21], [11], [24]. Column **Ref** documents the source for a value obtained from a previous paper. Character ▷ indicates that value originates from this paper. <sup>†</sup>indicates that network weights have been rescaled with [21] before fine-tuning, as is standard practice.

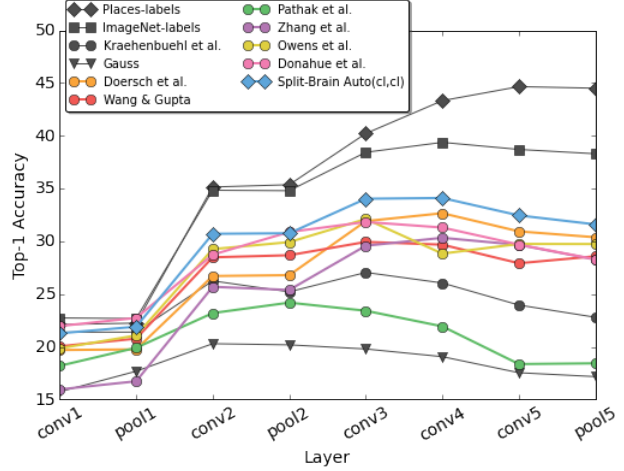
#### 4.1.2 Split-Brain Autoencoder Performance

Our primary result is that the proposed method, **Split-Brain Auto (cl,cl)**, achieves state-of-the-art performance on almost all established self-supervision benchmarks, as seen in the last row on Tables 2, 3, 4, over previously proposed self-supervision methods, as well as our ablation baselines. Figures 3(a) and (b) shows our split brain autoencoder method compared to previous self-supervised methods [4, 38, 28, 42, 5, 27] on the ImageNet and Places classification tests, respectively. We especially note the straightforward nature of our proposed method: the network simply predicts raw data channels from other raw data channels, using a classification loss with a basic 1-hot encoding scheme.

As seen in Figure 4(a) and Table 2, the autoencoder objective by itself, **Lab**→**Lab**, does not lead to a strong representation. Performance is near Gaussian initialization through the initial layers, and actually falls below in the conv5 layer. Dropping 50% of the data from the input randomly during training, **Lab(drop50)**→**Lab**, in the style of denoising autoencoders, adds a small performance boost of approximately 1%. A large performance boost is observed by adding a split in the architecture, **Split-Brain Auto (reg,reg)**, even with the same regression objective. This achieves 5% to 20% higher performance throughout

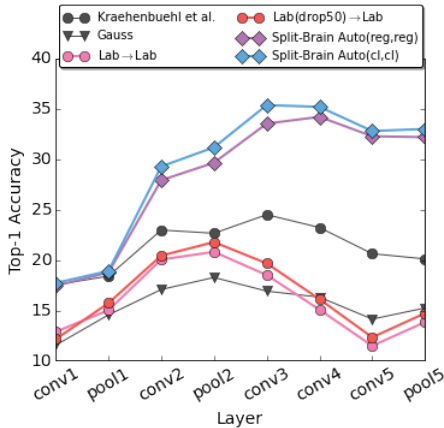


(a) ImageNet Classification

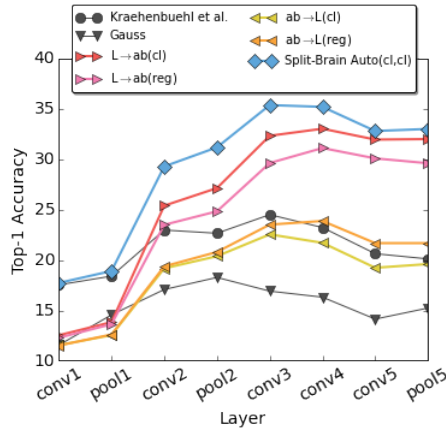


(b) Places Classification

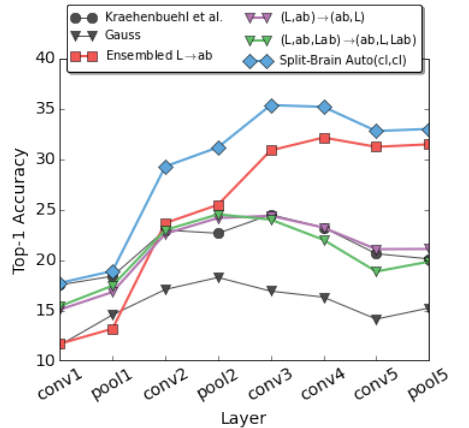
Figure 3: **Comparison to Previous Unsupervised Methods** We compare our proposed Split-Brain Autoencoder on the tasks of (a) ImageNet classification and (b) Places Classification. Note that our method outperforms other large-scale unsupervised methods [4, 38, 28, 42, 27, 5] on all layers in ImageNet and from conv2-5 on Places.



(a) Autoencoder Objective



(b) Cross-Channel Encoders



(c) Aggregation Methods

Figure 4: **Ablation Studies** We conduct various ablation studies on our proposed method, using the ImageNet classification benchmark proposed in [42]. Specifically, we compare (a) variations using an autoencoder objective (b) different cross-channel problems and loss functions (c) different methods for aggregating multiple cross-channel encoders.

the network, state-of-the-art compared to previous unsupervised methods. A further boost of approximately 1-2% throughout the network observed using a classification loss, **Split-Brain Auto (c1,c1)**, instead of regression.

#### 4.1.3 Cross-Channel Encoding Objectives

Figure 4(b) compares the performance of the different cross-channel objectives we tested on the ImageNet classification benchmark. As shown in [42] and further confirmed here, colorization,  $L \rightarrow ab(c1)$ , leads to a strong representation on classification transfer tasks, with higher perfor-

mance than other unsupervised representations pre-trained on ImageNet, using inpainting [28], relative context [4], and adversarial feature networks [5] from layers from conv2 to pool5. We found that the classification loss produced stronger representations than regression.

Interestingly, the task of predicting grayscale from color can also learn representations. Though colorization lends itself closely to a graphics problem, the application of grayscale prediction from color channels is less obvious. As seen in Tables 2 and 3 and Figure 4(b), grayscale prediction objectives  $ab \rightarrow L(c1)$  and  $ab \rightarrow L(reg)$  can learn representations above the **Gaussian** baseline. Though the learned



representation by itself is weaker than other self-supervised methods, the representation is learned on  $a$  and  $b$  channels, which makes it complementary to the colorization network. For grayscale prediction, regression results in higher performance than classification. How to choose the appropriate loss function for a given channel prediction problem is an open problem. However, it is interesting to note that the performance difference is typically small, which indicates that the cross-channel prediction problem is often times an effective method, even without careful engineering of the objective.

#### 4.1.4 Cross-Channel Encoder Aggregation Analysis

In Figure 4(c), we show variations on aggregated cross-channel encoders. To begin, we hypothesize that the performance improvement of split-brain autoencoders **Split-Brain Auto (cl,cl)** over single cross-channel encoders  $L \rightarrow ab$  is due to the merging of complementary signals, as each sub-network in **Split-Brain Auto** has been trained on different portions of the input space. However, the improvement could be simply due to an ensembling affect. To test this, we train a split-brain autoencoder, comprising of two  $L \rightarrow ab$  networks, **Ensemble  $L \rightarrow ab$** . As seen in Figure 4(c) and Table 2, the ensembled colorization network achieves lower performance than the split-brain autoencoder, suggesting that concatenating signals learned on complementary information is beneficial for representation learning.

We find that combining cross-channel encoders through concatenation is effective. We also test alternative aggregation techniques. As seen in Figure 4(c), training a single network to perform multiple cross-channel tasks  $(L,ab) \rightarrow (ab,L)$  is not effective for representation learning on full  $Lab$  images. Adding in the autoencoder objective during training,  $(L,ab,Lab) \rightarrow (ab,L,Lab)$ , in fact lowers performance in higher layers.

Our proposed methods outperform these alternatives, which indicates that (i) our choice of aggregating complementary signals improves performance (ii) concatenation is an appropriate choice of combining cross-channel encoders.

## 4.2. Split-Brain Autoencoders on RGB-D

We also test the split-brain autoencoder method on on registered images and depth scans from NYU-D [33]. Because RGB and depth images are registered spatially, RGB-D data can be readily applied in our proposed framework.

**Dataset & Detection Testbed** The NYUD dataset contains 1449 RGB-D labeled images and over 400k unlabeled RGB-D video frames. We use 10k of these unlabeled frames to perform unsupervised pre-training, as extracted from [14]. We evaluate the representation on the 1449 labeled images for the detection task, using the framework proposed in [14]. The method first trains individual detectors on the RGB and D domains, using the Fast R-CNN

Method	Data	Label	RGB	D	RGB-D
Gupta et al. [14]	1M ImNet [31]	✓	27.8	41.7	47.1
Gupta et al. [13]	1M ImNet [31]	✓	27.8	34.2	44.4
Gaussian	None	–	–	28.1	–
Krähenbühl et al. [21]	20 NYU-D [33]	–	12.5	32.2	34.5
Split-Brain Autoencoder	10k NYU-D [33]	–	<b>18.9</b>	<b>33.2</b>	<b>38.1</b>

Table 5: **Split-Brain Autoencoder Results on RGB-D images** We perform unsupervised training on 10k RGB-D keyframes from the NYU-D [33] dataset, extracted by [14]. We pre-train representations on RGB images using  $\ell_2$  loss on depth images in  $HHA$  space. We pre-train  $HHA$  representations on  $L$  and  $ab$  channels using  $\ell_2$  and classification loss, respectively. We show performance gains above Gaussian and Krähenbühl et al. [21] initialization baselines. The methods proposed by Gupta et al. [13, 14] use 1.3M labeled images for supervised pre-training. We use the test procedure from [14]: Fast R-CNN [11] networks are first trained individually in the RGB and D domains separately, and then ensembled together by averaging (RGB-D).

framework [11] on an AlexNet architecture, and then late-fuses them together through ensembling.

**Unsupervised Pre-training** We represent depth images using the  $HHA$  encoding, introduced in [13]. To learn image representation  $\mathcal{F}_{HHA}$ , we train an Alexnet architecture to regress from RGB channels to  $HHA$  channels, using an  $\ell_2$  regression loss.

To learn depth representations, we train an Alexnet on  $HHA$  encodings, using  $\ell_2$  loss on  $L$  and classification loss on  $ab$  color channels. We chose this combination, as these objectives performed best for training individual cross-channel encoders in the image domain. The network extracts features up to the `conv5` layer, using an Alexnet architecture, and then splits off into specific branches for the  $L$  and  $ab$  channels. Each branch contains AlexNet-type `fc6-7` layers, but with 512 channels each, evaluated fully convolutionally for pixel prediction. The loss on the  $ab$  term was weighted  $200\times$  with respect to the  $L$  term, so the gradient magnitude on the `pool5` representation from channel-specific branches were approximately equal throughout training.

Across all methods, weights up to the `conv5` layer are copied over during fine-tuning time, and `fc6-7` layers are randomly initialized.

**Results** The results are shown in Table 5 for detectors learned in RGB and D domains separately, as well as the ensembled result. For a Gaussian initialization, the RGB detector did not train using default settings, while the depth detector achieved performance of 28.1%. Using the stacked k-means initialization scheme from Krähenbühl et al. [21], individual detectors on RGB and D perform at 12.5% and 32.2%, while achieving 34.5% after ensembling. Pre-training with our method reaches 18.9% and 33.2% on the individual domains, above the baselines. Our RGB-D



ensembled performance was 38.1%, well above the Gaussian and Krähenbühl et al. baselines. These results suggest that split-brain autoencoding is effective not just on *Lab* images, but also on RGB-D data.

## 5. Discussion

We present split-brain autoencoders, a method for unsupervised pre-training on large-scale data. The split-brain autoencoder contains two disjoint sub-networks, which are trained as cross-channel encoders. Each sub-network is trained to predict one subset of raw data from another. We test the proposed method on *Lab* images, and achieve state-of-the-art performance relative to previous self-supervised methods. We also demonstrate promising performance on RGB-D images. The proposed method solves some of the weaknesses of previous self-supervised methods. Specifically, the method (i) does not require a representational bottleneck for training, (ii) uses input dropout to help force abstraction in the representation, and (iii) is pre-trained on full images, and thus able to extract features from the full input data.

An interesting direction of future work is exploring the concatenation of more than 2 cross-channel prediction sub-networks. Given a fixed architecture size, e.g. AlexNet, dividing the network into  $N$  disjoint sub-networks results in each sub-network becoming smaller, less expressive, and worse at its original cross-channel encoding task. To enable fair comparisons to previous large-scale representation learning methods, we focused on learning weights for a fixed AlexNet architecture. It would also be interesting to explore the regime of fixing the sub-network size and allowing the full network to grow with additional cross-channel encoders.

## Acknowledgements

This research was supported, in part, by NGA NURI, NSF SMA-1514512, an Intel research grant, and hardware donations by NVIDIA Corp and Algorithmia. We thank members of the Berkeley Artificial Intelligence Research Lab (BAIR), in particular Andrew Owens, for helpful discussions, as well as Saurabh Gupta for help with RGB-D experiments.

## References

- [1] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 37–45, 2015.
- [2] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016.
- [3] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.
- [4] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015.
- [5] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.
- [6] V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.
- [7] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.
- [8] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.
- [9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. <http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html>.
- [10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. <http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html>.
- [11] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.
- [12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.
- [13] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360. Springer, 2014.
- [14] S. Gupta, J. Hoffman, and J. Malik. Cross modal distillation for supervision transfer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [15] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.
- [16] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2016)*, 35(4), 2016.
- [17] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson. Learning visual groups from co-occurrences in space and time. *International Conference on Learning Representations, Workshop*, 2016.
- [18] D. Jayaraman and K. Grauman. Learning image representations tied to ego-motion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1413–1421, 2015.
- [19] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. *European Conference on Computer Vision*, 2016.

- [20] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *International Conference on Learning Representations*, 2014.
- [21] P. Krähenbühl, C. Doersch, J. Donahue, and T. Darrell. Data-dependent initializations of convolutional neural networks. *International Conference on Learning Representations*, 2016.
- [22] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [23] G. Larsson, M. Maire, and G. Shakhnarovich. Learning representations for automatic colorization. *European Conference on Computer Vision*, 2016.
- [24] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.
- [25] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. *European Conference on Computer Vision*, 2016.
- [26] A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. Conditional image generation with pixelcnn decoders. *arXiv preprint arXiv:1606.05328*, 2016.
- [27] A. Owens, J. Wu, J. H. McDermott, W. T. Freeman, and A. Torralba. Ambient sound provides supervision for visual learning. In *ECCV*, 2016.
- [28] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.
- [29] G. Ramanarayanan, J. Ferwerda, B. Walter, and K. Bala. Visual equivalence: towards a new standard for image fidelity. *ACM Transactions on Graphics (TOG)*, 26(3):76, 2007.
- [30] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text-to-image synthesis. In *Proceedings of The 33rd International Conference on Machine Learning*, 2016.
- [31] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.
- [32] R. Salakhutdinov and G. E. Hinton. Deep boltzmann machines. In *AISTATS*, volume 1, page 3, 2009.
- [33] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgb-d images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012.
- [34] P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, DTIC Document, 1986.
- [35] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.
- [36] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.
- [37] X. Wang, D. Fouhey, and A. Gupta. Designing deep networks for surface normal estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 539–547, 2015.
- [38] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2015.
- [39] X. Wang and A. Gupta. Generative image modeling using style and structure adversarial networks. *European Conference on Computer Vision*, 2016.
- [40] S. Xie and Z. Tu. Holistically-nested edge detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1395–1403, 2015.
- [41] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *International Conference on Learning Representations*, 2016.
- [42] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. *European Conference on Computer Vision*, 2016.
- [43] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014.